4d

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/827,971	04/06/2001	Jason Souloglou		5417

36183 7590 02/07/2006

PAUL, HASTINGS, JANOFSKY & WALKER LLP
P.O. BOX 919092
SAN DIEGO, CA 92191-9092

EXAMINER	
CHOW, CHIH CHING	

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 02/07/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 09/827,971	**Applicant(s)** SOULOGLOU ET AL.
	Examiner Chih-Ching Chow	**Art Unit** 2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 November 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *14 February 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *8/24/01, 3/17/03*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to REMARKS dated November 09, 2005.

2. Claims 1-18 remain pending.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1 and 17 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims **1 and 11** of copending Application No.09/828,049. Although the conflicting claims are not identical, they are not patentably distinct from each other, from the comparison listed in the following table:

Current Application (09/827,971) **US 2004/0205733A1**	**Co-Application (09/828,049)** **US 2002/0100030A1**
Claims	Claims

1. A method of generating an intermediate representation of program code, the method comprising the computer implemented steps of: (i)generating a plurality of register objects representing abstract registers, a single register object representing a respective abstract register; and (ii) generating expression objects each representing a different element of said program code as that element arises in the program code, each expression object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects.	1. A method for generating an intermediate representation of program code written for running on a programmable machine, said method comprising: (i) generating a plurality of register objects for holding variable values to be generated by the program code; and (ii) generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; wherein at least one variable sized register is represented by plural register objects, one register object being provided for each possible size of the variably sized register.
17. A system for generating an intermediate representation of program code, comprising: means for generating a plurality of register objects representing abstract registers, a single register object representing a respective abstract register; and means for generating expression objects each representing a different element of said program code as that element arises in the program code, each expression object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects.	10. A system for generating an intermediate representation of program code written for running on a programmable machine, the system comprising: means for generating a plurality of register objects for holding variable values to be generated by the program code; and means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; wherein at least one variably sized register is represented by plural register objects, one register object being provided for each possible size of the variably sized register.

5. Claim 1 of current application is anticipated by co-appliationclaim 1 in that co-application claim 1 contains all the limitations of the current application claim 1. Claim 1 of the current application therefore is not patentably distinct from co-application claim 1 and as such is unpatentable for obvous-type double patenting.

6. Claim 17 of current application is anticipated by co-appliationclaim 11 in that co-application claim 10 contains all the limitations of the current application claim 17. Claim

17 of the current application therefore is not patentably distinct from co-application claim 10 and as such is unpatentable for obvious-type double patenting.

7. This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Response to Arguments

8. Applicants' arguments for Claims 1-18 have been fully considered respectfully by the examiner but they are not persuasive.

9. Applicants' arguments are basically in the following points:

- **The Obviousness Rejection Over Aho In View of Koizumi**

Examiner's Response: The argument about Aho's intermediate representation is an intermediate step between the source program and the garget program (REMARKS, pages 3-7), claim 1 is not concerned with the second stage, naming generating target code – this argument is valid, however, Koizumi's teaching covers the first stage of the current application. See 35 USC § 102 and 35 USC § 103 Rejections below.

- **The term "Abstract Registers" in Koizumi** – see REMARKS dated 11/07/05 page 8

Examiner's Response: The argument about Koizumi's 'Abstract Registers' are not persuasive. Koizumi's disclosure covers two parts, the first part is translating the program source code to intermediate program, and using the abstract registers for storage, the second part is the translation from the intermediate program into a certain target computer. The first part of Koizumi's teaching covers the current application.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 1-4, 7-18 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,586,323 by Shinobu Koizumi et al. (hereinafter "Koizumi").

CLAIMS	**Koizumi**
1. A method of generating an intermediate representation of program code, the method comprising the computer implemented steps of: a. generating a plurality register objects representing abstract registers, a single register object representing a respective abstract register; and b. generating expression objects each representing a different element of said program code as that element arises in the program code, each expression object being referenced by a register object which it relates either directly, indirectly via references from other expression objects.	With respect to claim 1, item a, see Koizumi's column 1, lines 25-30, "On the other hand, in the case of the interpreter system, a language (referred to as the intermediate language) which differs from the machine language of the target computer (*intermediate representation of program code*) is prepared along with a program (referred to as the interpreter) which is adapted to interpret and execute the **intermediate language program** on the target computer." Also in Koizumi column 4, lines 53-58, "an **abstract register** machine (also referred to as ARM or Arm in abbreviation) having a plurality of registers is presumed, wherein an instruction sequence for the **abstract register** machine or ARM is made use of as a basic part of the common object program (referred to as the abstract object program)". For item b see Koizumi's Fig. 19, items 3956, 3958, the 'common equation' is the as the 'expression object', which is directly or indirectly via references from other expression objects; see Koizumi's column 22, lines 17-25, "Subsequently, a common expression or equation in the extended basic block is deleted. To this end, starting from the first extended **basic block** (step 3954), the common equation in the extended basic blocks is picked out (step 3956). The succeeding common equation is replaced by the content of a register or a variable representing the result of the common equation picked out preceding (step 3958).

	This procedure is repeated until the last extended basic block has been reached (steps 3960, 3962)."
2. A method according to claim 1 wherein said program code is expressed in terms of an instruction set of a subject processor.	For claim 1 feature see claim 1 rejection, for rest of claim 2 feature see Koizumi's Abstract, "A compiler translates a source program (*program code*) into an abstract object program including **an abstract machine instruction sequence** (*expressed in terms of an instruction set*) and indication concerning **allocation of abstract registers**. An installer converts the abstract object program into a machine language program of **target computer** on the basis of **executable computer** (*subject processor*) specification information including register usage indication and machine instruction selecting rules."
3. A method according to claim 2, wherein said register objects represent abstract registers corresponding to registers of said subject processor.	See claim 2 rejection.
4. A method according to claim 1, wherein each of said steps are performed sequentially for basic blocks of said program code having only one effective entry point instruction and one effective exit point instruction.	See claim 1 rejection about 'basic blocks'; Koizumi teaches a basic block has only one effective entry point instruction and one effective exit point instruction in an analogous art for the purpose of sending blocks of the original code to the compiler. In Koizumi, column 21, lines 59-62, "At first, an ArmCode instruction sequence of the ArmCode program is divided into a plurality of **basic blocks** by **punctuating the sequence at flow-in points and branching points** (*entry and exit point*) of the control".
7. A method according to claim 1, wherein a single said expression object is generated	For the feature of claim 1 see claim 1 rejection, for the rest of claim 7 feature see

for a given element of said program code, and each said expression object is referenced by said register objects to which relates.	Koizumi, column 22, lines 30-64, about abstract register assignment (eliminating duplications), specifically, lines 50-51, "in the instruction 4098, the abstract registers Ar6 and Ar11 are **reused**, while in the instruction 4076, the register Ar7 is reused with the register Ar5 being reused in the instructions 4108, 4110 and 4112, as shown in FIGS. 20 and 21", -- the abstract registers are reused for different expression objects after the optimization processing.
8. A method according to claim 1, wherein if a said register object or a said expression object becomes redundant or unnecessary it is eliminated.	For the feature of claim 1 see claim 1 rejection. For the rest of claim 8 feature see Koizumi column 22, lines 54-59, "In this manner, these abstract registers are replaced by those having the identical values (*redundant or unnecessary object*), respectively, whereby the instructions for determining the values of the abstract registers Ar8, Ar12, Ar15, Ar9, Ar13, Ar15 and Ar16 are **deleted**." (*redundant or unnecessary expression object is eliminated*)"
9. A method according to claim 8, wherein a redundant or unnecessary said register object or said expression object is identified by maintaining an ongoing count of references being made to that object as a network of register and expression objects is constructed.	For the feature of claim 8 see claim 8 rejection. For the rest of claim 9 feature see Koizumi column 13, lines 13-15, "function: "alloc" (abstract register name, register type, discriminant variable, instruction number, **preserve count**, priority)" – Koizumi keeps track of an ongoing count of references being made to an object.
10. The method according to claim 9, wherein for each expression object count is maintained of the number of references to that expression object from other expression objects or from register objects, the count associated with particular expression object being adjusted each time	See rejection of claim 8.

a reference to that expression object is made or removed.	
11. A method according to claim 10, wherein an expression object and all references from that expression object are eliminated when said count for that expression object is zero.	See rejection of claim 8. It's obvious that an object is eliminated if the reference count is zero.
12. The method of claim 1, comprising translating the program code written for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the generated intermediate representation.	For the feature of claim 1 see claim 1 rejection. For the rest of claim 12 feature see Koizumi's column 1, lines 31-35, "the high-level language program is translated into the intermediate language program which is then executed by the target **computer** or machine on **which the interpreter program runs**." Also see column 2, lines 63-67, "In order to allow a machine-independent intermediate language program (i.e. intermediate language program which is independent of any specific target machine or computer) to be adopted as a form for preservation and management of a program to be executed repeatedly".
13. The method claim 12, wherein said translating step is performed dynamically as the program code is run	See claim 12 rejection, translation is performed dynamically as the program is run.
14. The method of claim 1, comprising optimising the program code by optimizing the generated intermediate representation.	For the claim 1 feature see claim 1 rejection. For the rest of claim 14 feature see Koizumi's column 3, lines 11-15, "In the course of translation or conversion of the intermediate language program into the machine language program, **optimization of the program** is carried out by taking into consideration the characteristics of the target computer which is to execute that program.".

15. The method of claim 14, wherein said optimizing step is used to optimise the program code written for execution by a processor a first type so that the program code may be executed more efficiently by that processor.	For the feature of claim 14 see rejection of claim 14. See Koizumi, column 22, lines 30-64, about abstract register assignment (eliminating duplications), specifically, lines 50-51, "in the instruction 4098, the abstract registers Ar6 and Ar11 are **reused**, while in the instruction 4076, the register Ar7 is reused with the register Ar5 being reused in the instructions 4108, 4110 and 4112, as shown in FIGS. 20 and 21", -- the abstract registers are reused for different expression objects after the optimization processing.
16. A method for generating an intermediate representation of program code written for running on programmable machine, said method comprising: (i) generating plurality of register objects for holding variable values to be generated by the program code; and (ii) generating plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; said objects being organized into a branched tree-like network having all register objects at the lowest basic root or tree-trunk level of the network with no register object feeding into any other register object.	For items (i) and (ii), see Koizumi column 14, lines 35-60. Koizumi's disclosure teaches using plurality of register objects for holding variable values, and relationships between said fixed values and said variables (position of a variable name in a symbol table or abstract register number representing an address in the memory), for the tree-like network see claim 1 rejection.
17. A system for generating an intermediate representation of program code, comprising: means for generating a plurality of register objects representing abstract registers, a single register object representing a respective abstract register; and	See rejection of claim 1.

means for generating expression objects each representing a different element of said program code as that element arises the program code, each expression object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects.	
18. A system for generating an intermediate representation of program code written for running on a programmable machine, the system comprising: means for generating a plurality of register objects for holding variable values to be generated by the program code; and means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; wherein said objects are organised into a branched tree-like network having register objects the lowest basic root or tree-trunk level of the network with no register object feeding into any other register object.	See rejections of claim 16.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 5, 6 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,586,323 by Shinobu Koizumi et al. (hereinafter "Koizumi"), in view of Aho

et al, "Compiler, principles, techniques, and tools" book, published in 1986 (herein after "Aho").

CLAIMS	**Koizumi / Aho**
5. A method according to claim 1, wherein at least some said expression objects feed into more than one said register object.	For the feature of claim 1 see claim 1 rejection. Koizumi teaches all aspects of the applicant's claims but it does not specifically mention 'expression objects feed into more than one register objects". However, Aho teaches it in an analogous prior art. See Aho, page 559, Fig. 9.20, each of the t2, t3, t1 and t4 are 'expression objects' and they are feed into more than one register object. (E.g. t1 is an expression object, it feed into register objects a and b; t2, is also an expression object, it feed into register objects c and d). Any program would have some expression objects since the program should perform certain functions, functions are 'operations' and they are represented by 'expression objects'; operands feed into operations, here operands are represented by 'register objects'. It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **intermediate representation** for abstract registers of Koizumi with the feeding expression objects into more than one register objects taught by Aho, for the purpose of performing computation for a program (See Aho, page 559, 2nd paragraph).
6. A method according to claim 1, wherein said expression objects are not duplicated.	For the feature of claim 1 see claim 1 rejection. In Aho page 290, under 'Directed Acyclic Graphs for Expressions' section, "A directed acyclic graph (hereafter called a dag) for an expression identifies the common subexpressions in the expression.

Like a syntax tree, a dag has a node for every subexpression of the expression; an interior node represents an operator and its children represent its operands." In addition, on page 291, first paragraph, "A dag is obtained if the function constructing a node first **checks to see whether an identical node already exists**. ... if so, mknode can return a pointer to the previously constructed node" – the duplication of a function/operation node (expression object) is checked before a new function node is created.

Conclusion

The following summarizes the status of the claims:

35 USC § 102 rejection: Claims 1-4, 7-18

35 USC § 103 rejection: Claims 5, 6

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow

Examiner

Art Unit 2192

February 2, 2006

cc

[illegible]

TUAN DAM
SUPERVISORY PATENT EXAMINER